CSE 351 Section 8 – Virtual Memory

Hi there! Welcome back to section, we’re happy that you’re here ☺

Address Translation

Address Translation Symbols

Basic Parameters:

\[ N = 2^n \]  
Number of addresses in virtual address space

\[ M = 2^m \]  
Number of addresses in physical address space

\[ P = 2^p \]  
Page size (bytes)

Components of Virtual Address (VA):

- **VPO**: Virtual Page Offset
- **VPN**: Virtual Page Number
- **TLBI**: TLB Index
- **TLBT**: TLB Tag

Components of Physical Address (PA):

- **PPO**: Physical Page Offset (Same as VPO)
- **PPN**: Physical Page Number

![Address Translation Diagram](image_url)
A Small Example

Suppose we have a simple memory system with **14-bit** virtual addresses, **12-bit** physical addresses, and a page size of **64 bytes**. The TLB has **16 entries** in total and is **4-way** set associative. The cache is direct-mapped with **16 sets** and a block size of **4 bytes**.
Memory Requests

1. Virtual Address: 0x03D4

   **Physical Address:**

   CT ____  CI ____  CO ____  Cache Hit? ____  Data (byte) ____

2. Virtual Address: 0x038F

   **Physical Address:**

   CT ____  CI ____  CO ____  Cache Hit? ____  Data (byte) ____
3. **Virtual Address: 0x0020**

![Diagram of TLB and physical address mapping]

VPN ___ TLBT ___ TLBI ___ TLB Hit? ___ Page Fault? ___
PPN ___

**Physical Address:**

![Diagram of CT, CI, CO mapping]

CT ___ CI ___ CO ___ Cache Hit? ___ Data (byte) ___

4. **Virtual Address: 0x036B**

![Diagram of TLB and physical address mapping]

VPN ___ TLBT ___ TLBI ___ TLB Hit? ___ Page Fault? ___
PPN ___

**Physical Address:**

![Diagram of CT, CI, CO mapping]

CT ___ CI ___ CO ___ Cache Hit? ___ Data (byte) ___
Virtual Memory Table

<table>
<thead>
<tr>
<th>VA width (n)</th>
<th>PA width (m)</th>
<th>Page size (P)</th>
<th>VPN width</th>
<th>PPN width</th>
<th>Bits in PTE (assume V, D, R, W, X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>32</td>
<td>16 KiB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>26</td>
<td></td>
<td></td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td>21</td>
<td></td>
<td>22</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>32 KiB</td>
<td>25</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
<td>48</td>
<td></td>
<td>29</td>
</tr>
</tbody>
</table>

Fork and Concurrency

Consider this code using Linux’s `fork`:

```c
int x = 7;
if( fork() ) {
    x++;
    printf(" %d ", x);
    fork();
    x++;
    printf(" %d ", x);
} else {
    printf(" %d ", x);
}
```

What are all the different possible outputs (i.e. order of things printed) for this code?

(Hint: there are four of them.)